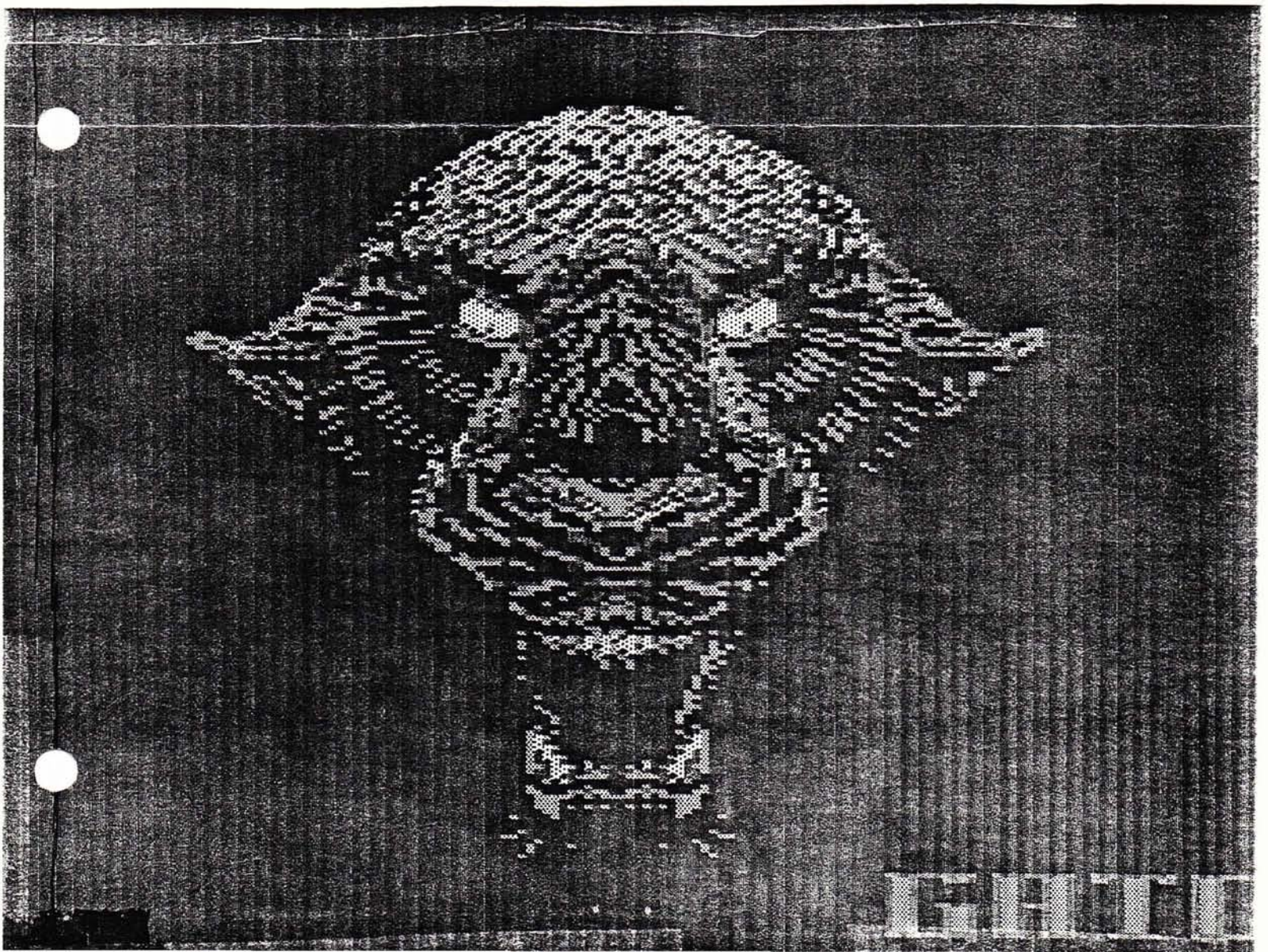
O-ACE'S

Omaha Atari Computer Enthusiasts

Volume 5 Issue 8 August 1987

Wednesday, AUGUST 12
MONTHLY MEETING
La Vista Recreation Center
7:00 P.M.

THERE WILL BE NO SIG MEETINGS UNTIL FURTHER NOTICE



OLD BUSINESS

The summer time lull is here again and was especially evident at the July meeting as only a small turnout of members were present. Because of the small turnout and lack of activity of major computer events, the meeting which began at 7:45 was ajourned at 8:00 with a only a few topics discussed.

As usual, the first topic of the meetings is the monthly door prize drawings, and this months winners were Frank Babka and Jim Herrington. Following this, the July newsletter was reviewed after which an update on the 8-bit library was given. The clubs programming contest was also discussed once again.

DEMONSTRATIONS

After the general meeting, the club was given a demonstration of Xanth's Midi Maze game. Midi Maze is a multi-player ST video game that allows up to 16 ST computers to hook together through the midi port to allow each player to compete against one another in a shoot and destroy maze game. Although only two machines were hooked together at the July meeting, the game was still fun to play and watch.

SUB GROUP MEETINGS

Although not offically sponsered by our club(O-ACE), there is a get together of Atari users every other Sunday at Offutt Air Force Base (SAC). These meetings occur at the Recreation Center from 3:00-7:00PM.

The next meeting will take place August 16 and will be a combination picnic/garage sale. Everyone is invited to bring along food and drink and any computer related items they wish to sell to the next meeting.

For more information contact Lance Summers at 291-4704 or Deane Bolin at 291-1678.

MEMBERSHIP INFORMATION

<u>September Expirations:</u> Jamie Blinn, Glen Flint, Bruce Harvey, Theodore Henderson, and Dolly Heyden

NOTES FROM THE PRESIDENT

By Jamie Blinn

This article should fill our members in on what's in store for the club in the coming months.

Hopefully, the lack in attendance at current club meetings can be attributed to the great weather we've been having. As school starts for the unfortunate and weather begins to go sour, it's time to find some indoor activities. With this in mind and with some of the changes I am going to try and make, this would be a great time to get back with the club again.

Remember, the more people that join the club the better quality meetings and services we can provide. As further incentive I want to be able to lower the dues. I feel that the club can still provide for itself if the dues were lowered to $18. In order to be fair I feel that we should extend all existing memberships by 2 months. The last people we should hurt are the people that have been with the club the longest. Of course, I can't make this change by myself and we will probably be discussing it at the next meeting. This is a good chance to express your opinions.

Onto the next subject. I've talked to the president of LAUGON (Lincoln User Group) about combining newsletters. This would allow for a larger newsletter, more original articles and a cut in cost. I won't go into the specifics right now, but the same newsletter would be sent to members of both groups. One or t pages would be group specific for Omaha and Lincoln. The rest of the newsletter would be articles from our members, their members and newsletter exchanges.

The LAUGON president was pleased with the idea and he is going to discuss it with their newsletter editor. Along the same lines we are now going to begin to seek advertising for our newsletter. If you have any ideas on who we might want to contact to advertise, talk to me at one of the meetings.

The last major thing I should mention concerns our ST members. Our ST library has been growing rapidly since it was first started, but it still is not up to par with our 8-bit library. We have a chance to buy 100 Public Domain ST disks. At this time I am not sure whether the club can foot the bill for the disks. I feel that the club can pay for at least half, if not more. The other half would work the same way as the Antic Archives did several years ago. Everyone who wanted copies had to pay a certain amount and then were entitled to them. If the club can not pay for the entire purchase we will have a sign-up list for anyone interested. After we see how many people sign-up we will decide the cost per member. It will not exceed $10. This does not include the $2 for each disk.

I guess I have rambled enough for now. As I said before, we will probably be discussing most of this at the August meeti so plan on attending.

Hints, tricks, tips, etc.

By Steven Ourada

Player/Missle Graphics

Although there are many very good tutorials on this subject in various magazines and books, I believe there are still many who cannot find adequate information about it.

First, the definition: Players are vertical strips of the Atari's screen, 8 pixels wide, which can be easily moved around the screen and do not erase the underlying display. These features make them ideal for animation and special graphics applications. Missles are the same as players, except they are only 2 pixels wide. These are usually used for lasers, bullets, etc. fired from the players.

Now, let's get into the technical stuff. In order to use player/missle graphics (P/M for short), you must first set aside a portion of memory for that use. The size of the portion depends on whether you want to use single-line or double-line resolution. In single line resolution, each line of the players/missles is one scan line high on the monitor/TV. With double line resolution, you guessed it, each line of the player is two lines high. If you need single-line, you must reserve 2048 bytes of memory. The reserved memory must also start on a 2K boundary, such as 16K (16,384) or 18K (18,432). If you plan to use double-line resolution, you must reserve 1024 bytes of RAM, and it must begin on a 1K boundary.

The first 384 bytes in double, or 768 bytes in single resolution, is wasted. Well, it isn't used by the player/missle graphics, but it can be used for other things, such as storage of other player images. After that, the next 128 bytes in double, and 256 in single, is the definition for the missles. Each byte in the definitions contains the data for one line of each of the four missles. Here's a little chart to help:

```
+BIT0-1+BIT2-3+BIT4-5+BIT6-7+
|  M3  |  M2  |  M1  |  M0  |
+------+------+------+------+
```

The next 512 bytes double/1024 single define the players. The definitions are in sequential order from 0-3, each player having 128 (double) or 256 (single) bytes.

Colors for the players/missles are selected through bytes 704-707. The values must be poked in if you are using Atari Basic. The values are the standard ones explained in the Atari Basic Manual. Each player/missle pair has its own color. (The missles may be given a common color, I'll explain that later.)

Horizontal movement of players/missles is easy. With one simple poke, (locations 53248-53255, players 0-3, missles 0-3, respectively) they are instantly moved to the position chosen. This can even occur more than once in the scan of the screen, so by using display list interrupts, one can use one player for more than one independent (sort of) object. Horizontal positions range from 0-255, although not all are visible.

Vertical movement of P/M graphics is not directly supported by the hardware, so it is much harder to implement. In order to obtain vertical movement, you must move the bytes which define the object to a different place in the memory reserved to define the players. Because of the large number of bytes to be moved for most objects, it is usually impractical to use Basic for such tasks. Most often, machine language code is used to move the objects within the players/missles. It is beyond the scope of this article to present such a routine, but many can be found in magazines such as Analog, Antic, and Compute.

P/M graphics also have separately definable widths. Each player's width is defined by bytes 53256-53259 (players 0-3, respectively). A zero in these locations means normal width, a one means double, and a three means quadruple.

Sixteen bytes (from 53248-53263) are called the collision registers. They record any overlapping of player/missles and other players/missles or playfield (normal pixels on the screen, such as those formed with PLOT or DRAWTO) areas. Here is a chart to show the values in these bytes after collision:

P=player, M=missle, C=color.

Memory loc.	P0	P1	P2	P3
P0/53260	X	2	4	8
P1/53261	1	X	4	8
P2/53262	1	2	X	8
P3/53263	1	2	4	X

Memory loc.	C0	C1	C2
P0/53252	1	2	4
P1/53253	1	2	4
P2/53254	1	2	4
P3/53255	1	2	3

Memory loc.	C0	C1	C2
M0/53248	1	2	4
M1/53249	1	2	4
M2/53250	1	2	4
M3/53251	1	2	4

Memory loc.	P0	P1	P2	P3
M0/53256	1	2	4	8
M1/53257	1	2	4	8
M2/53258	1	2	4	8
M3/53259	1	2	4	8

(The above chart was taken from Compute!'s First Book Of Atari Graphics [I recommend this book highly.])

When a collision register is set, it remains set until 53278 is poked with any value. This clears all collision registers.

When players overlap on the screen, the computer obviously can't display both of them in the same place at the same time. (Sounds like a physics class, doesn't it?) So, you must tell the computer beforehand which of the players or playfield colors is the most important. How do you tell it? By poking location 53275 with the proper value. This location is called the priority register. It's laid out like this:

+BIT0-3	+BIT4	+BIT5	+BIT6-7	+
¦Priority	¦Fifth	¦Multi-	¦GTIA	¦
¦Select	¦Player	¦Color	¦Display	¦
¦	¦Enable	¦Players	¦Selects	¦

Now a little explanation: The priority select bits are used to tell the computer about which players and colors are most and least important. Bit 0 selects: all players, then all playfields. Bit 1: Players 0-1, playfields, players 2-3. Bit 2: Playfields, players. Bit 3: Playfield colors 0-1, players, colors 2-3. If more than one bit is selected, any overlapped objects with conflicting priorities will turn black in the overlapping area.

The fifth player enable does just what it says. Almost. It makes all the missles change to the color in color register 3, instead of their player color. Using this, you can move all 4 missles next to each other to simulate a fifth player.

The multi-color players bit allows the overlap region between players 0 and 1 and players 2 and 3 to become yet another color. I have no information on what that color is. I guess you just have to experiment.

The GTIA select bits are for enabling graphics modes 9-11.

+BIT6	+BIT7	+Mode +
¦ 0	¦ 0	¦Non-GTIA ¦
¦ 0	¦ 1	¦GTIA 09¦
¦ 1	¦ 0	¦GTIA 10¦
¦ 1	¦ 1	¦GTIA 11¦

Three last things to say:

1) To enable P/M graphics, put a 3 in loc. 53277. To disable them, put a 0 there.

2) Before enabling them, put the high byte (address/256) of the beginning of the memory set aside for P/M graphics.

3) In location 559, put a 62 for single line, or a 46 for double line, resolution.

Well, I hope I covered about everything you need to know. If you have any more questions about P/M graphics, feel free to ask. Also, if you have any ideas for future 'Hints, Tricks, Tips, etc.' articles, please tell me about it.

Thanks for reading!

Yamaha FB-01
FM Sound Generator

Reviewed by Deane Bolin

Tired of everyday, ho-hum music on your ST? Ears still ring from that tinny monitor speaker? Would you like to blast your ears with a piece from Mozart, or one of the great masters? Or how about listening to 'On Broadway' in eight parts at once. In whatever type instrument you choose? Then wait no more.

Now, there is the Yamaha FB-01. A quality product that you can afford and will let you do just about anything you want to do with music and your ST. Or your 8-bit if you have a MIDI interface.

The FB-01 is an FM sound generator with a four operator, 8-algorithm FM tone generator that can produce up to eight notes simultaneously. You may set the FB-01 to be eight instruments, each playing in mono mode, or one instrument with eight notes in poly mode, or any combination in between. But beware, you only have 240 voices built in, and only 96 voice memories in RAM for you to use for making your own. So if you think you'll be using more than this, you'll really have to spend money for the big time stuff. But really, 240 built in voices and 96 user programable voices in RAM should be all you should ever need unless you are going into big time productions.

All settings for each instrument, MIDI channels, voice numbers, settings, etc., can all be stored in Configuration Memory. There are sixteen for your own use and four ROM presets.

The FB-01 comes with a MIDI In, MIDI Out, and MIDI Thru jacks. There are also left and right jacks for the audio out. You will need some type of amplifier for your output. You can plug the FB-01 into either one or two guitar type amps, a portable stereo system, or your home stereo system. If you use only one amp, you can use a spliter to run left and right sides together. But in so doing, you lose the stereo capability of the FB-01. The jacks are the standard 1/4 inch jacks.

Built into the system is also a five year battery to back up all of your configurations. This must be replaced by a Yamaha service dealer.

Set up and operation are pretty straight forward. The manual is well written and has lots of examples, but it expects you to be familiar with your own MIDI device, and doesn't go into any computers or keyboards for you. All functions are controlled from the front panel with eight pushbutton switches. This doesn't seem like much, but with these eight buttons(multi-function) you can totally control the sounds you want.

Following is a short rundown of the control buttons and some of their functions:

1. System Set Up:
   a. select the configuration you want to use.
   b. Combine on/off to save info with each voice.
   c. Memory protect allows you to store to RAM.
   d. Configuration store lets you store you own configurations
   e. Tuning lets you tune up or down two octaves
   f. System Channel Number for System Exclusive messages
   g. Dump lets you system or MIDI dumps
   h. Keycode Number Receive Mode

2. Instrument Select: increments instrument number

3. Data entry/-1/No: decreases value

4. Data entry/+1/No: increases value

5. Inst Assign:
   a. select MIDI Channel Number
   b. set number of notes
   c. set key-code Number limit/Lower
   d. set Key-code Number limit/High

6. Inst Function:
   a. set output level for each instrument
   b. set octave transpose up or down two octaves
   c. detune each instrument up or down
   d. set each channel or instrument for Left, Right, or Both
   e. set LFO enable on or off

7. Voice Function:
   a. set pitchbender range
   b. set portamento time
   c. set either poly or mono mode
   d. set the controller for PMD

8. Voice Select:
   a. select/set voice number
   b. select/set voice store
   c. select/set voice bank number

Creating new voices or modifying voices on the FB-01 requires a computer and the appropriate software. Some of the computers commonly used with the Yamaha FB-01 are: Apple II series and Mac, Atari series, Commodore series, IBM and compatables, and the Yamaha CX5M.

Configuration can be set any way you like, as long as you don't exceed the eight note limit. You can set it for one instrument to play eight notes, two instruments to play four note, four to play two each, or any combination you choose. Then you may set one to play on the left and one on the right and one on both as you like. Each instruments's volume may be set as you like.

All in all, I have found the Yamaha FB-01 FM generator to be a top notch, first class musical instrument that can add many hours of musical enjoyment. You can use it with most of the music programs that are available for the ST and 8-bit machines. It has worked well with Music Studio, Midi-Play, Ez-Track and a couple of others that I have personally tried.

The only drawback that I was able to find, and this may only be a personal thing, was that you cannot go directly to the voice you want to use, but must use the increment or decrement buttons to get there. But since once you get things set up, you don't have to change them anymore, it really isn't a problem. And for a price of only $300 locally, this is a deal that's hard to pass up. Especially if you don't play a keyboard(like myself) but do like high quality music.

THE ASSEMBLER LANGUAGE COURSE PART 7

As you will see, the additions to the ADDRESS program presented last month are not hard but are strung out over the entire code structure.

First plan out what you intend on doing. My way is to go to the flow and modify it for the inclusion to the read from disk test.

The flow now reads like this:

```
2: Flow the task.
   A: Initialize Variables
   B: Ask for file name to save to.
   C: Open Printer and File
   D: If Opens are OK
      1) Loop Until NAME$ = -END-
         a) Clear screen
         b) If DVC = keyboard
            1] Print to the screen:
               a] ENTER NAME(LAST, FIRST MI)
         c) Read into NAME$ from DVC
         d) If NAME$ <> -END-
            1] If NAME$ <> -DISK-
               a] If DVC = keyboard
                  1> Print to the screen:
                     a> ENTER ADDRESS
                  (STREET, CITY, STATE, ZIP)
               b] Read into ADDRESS$ from
                  DVC
               c] If DVC = keyboard
                  1> Print to the screen:
                     a] ENTER PHONE NUMBER
                        ((###) ###-####)
               d] Read into PHONE$ from DVC
            2] ELSE
               a] Print to screen:
                  a> ENTER FILE NAME TO READ
                     FROM
               b] Read into FILENAME$
               c] Open FILENAME$
               d] If open is ok
                  a> DVC = DISK
                  b> Read into NAME$ from
                     disk
                  c> If NAME$ <> -END-
                     1> Read into ADDRESS$
                        from disk
                     2> Read into PHONE$
                        from disk
         e) Clear screen
         f) Print NAME$
            1] To Drive
            2] If NAME$ <> -END-
               a] To Screen
               b] To Printer
```

*** CLUB OFFICERS ***

President		- Jamie Blinn/592-0918
Vice Pres		- Peter Killian/592-5427
Librarian	8-Bit	- Axel Ricker/330-7734
	ST	- Donna Griggs/455-8317
Editor	8-Bit	- Roger Reese/331-1336
	ST	- Deane Bolin/291-1678
Member At Large		- Vacant

WE WILL BE MEETING AT THE LA VISTA RECREATION CENTER ON WEDNESDAY AUGUST 12

The Omaha Atari Computer Enthusiasts (O-ACEs) will now be meeting on the second Wednesday of the month. Our meeting loation is the La Vista Recreation Center located at 8116 Park View Blvd., behind the La Vista City Hall and Police Station. Park View Blvd. is the next street south of Harrison when approaching from 84th street. Meeting time is 7:00, with the general meeting beginning at 7:30.

O-ACES
P.O. BOX 723
PAPILLION NE. 68046

S.L.C.C
P.O. BOX #1506
SAN LEANDRO CA, 94577

```
            c] Print ADDRESS$
               1> To Drive
               2> To Screen
               3> To Printer
            d] Print PHONE$
               1> To Drive
               2> To Screen
               3> To Printer
      2) end loop for NAME$ = -END-
   E: Close Printer, Write file, and Read file
```

And now for the changing of the Assembler code.

```
BETWEEN LINE 0230 AND 235 INSERT
            0232     STX DVC

BETWEEN LINE 0485 AND 0490 INSERT
            0487 RA3 LDX DVC
            0488     BNE R3A

CHANGE LINE 0490 RA3 LDY #S2-SA
TO          0490     LDY #S2-SA

CHANGE LINE 0510     LDA #21
TO          0510 R3A LDA #21

CHANGE LINE 0540     JSR RD0
TO          0540     JSR RD

CHANGE LINE 0680 RA5 LDY #S3-SA
TO          0680     LDY #S3-SA

CHANGE LINE 0700     LDY #LN1-SA
TO          0700 R5C LDY #LN1-SA

CHANGE LINE 0730     JSR RD0
TO          0730     JSR RD

BETWEEN LINE 0750 AND 0755
INSERT      0752     LDX DVC
            0753     BNE R5D

CHANGE LINE 0780     LDY #LN2-SA
TO          0780 R5D LDY #LN2-SA

CHANGE LINE 0810     JSR RD0
TO          0810     JSR RD

BETWEEN LINE 0830 AND 0835
INSERT      0832     LDX DVC
            0833     BNE R5E

CHANGE LINE 0860     LDY #PHN-SA
TO          0860 R5E LDY #PHN-SA

CHANGE LINE 0890     JSR RD0
TO          0890     JSR RD

BETWEEN LINE 1760 AND 1765
INSERT      1762 DRN .WORD DRP,LDRP
            1763 DNR .WORD RD2,LRD2

BETWEEN LINE 1860 AND 1865
INSERT      1861 DRP .BYTE "ENTER FILE
                 NAME TO SAVE TO:",$9B
            1862 LDRP = *-DRP
            1863 RD2 .BYTE
                 "D2:FILENAME.EXE",$9B
            1864 LRD2 = *-RD2

BETWEEN LINE 1920 AND 1930
INSERT      1925 DSK .BYTE "-DISK-",$9B

ADD LINE    1950 DVC .BYTE 0
```

NOW FOR A LITTLE TRICK, YOU WILL HAVE TO RENUMBER YOUR PROGRAM ABOUT THREE TIMES TO INSERT THE FOLLOWING LINES BETWEEN LINE 0670 AND LINE 0675.

THE LINES TO INSERT ARE:

```
0001 RA5 LDY #0
0002 R5A LDA NA,Y
0003     CMP DSK,Y
0004     BNE R5B
0005     INY
0006     CPY #7
0007     BNE R5A
0008     LDY #DRN-SA
0009     JSR PR0
0010     LDA #$10
0011     STA DNR+2
0012     LDY #DNR-SA
0013     JSR RD0
0014     LDX #$40
0015     JSR CL
0016     LDA #4
0017     LDY #DNR-SA
0018     JSR OPN
0019     LDA $343,X
0020     BPL R5R
0021     LDX #0
0022 R5R STX DVC
0023     JMP RA3
0024 R5B LDX DVC
0025     BNE R5C
```

Ok, that was not so hard. So for next month, try to figure out a way to find a given name from your database on disk. And for those of you who are beginning to catch on to this game of explicit commands, try to add a sort to the program.

Until next month, Enjoy, Lance Summers...